It's all multimedia now! (so what's next?)

Presentation by Stephen Travis Pope
stephen@HeavenEverywhere.com
YOW 2014, Dec, 2014

Overview

Introduction

Historical Perspective

Past, Present & Future

Services, Networks & Tools

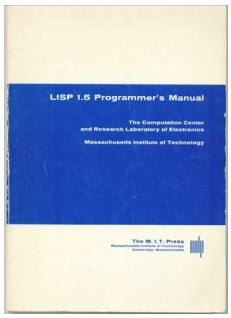
(Slides on the web @ HeavenEverywhere.com)

Introduction

WhoAmI ?

The Big Picture

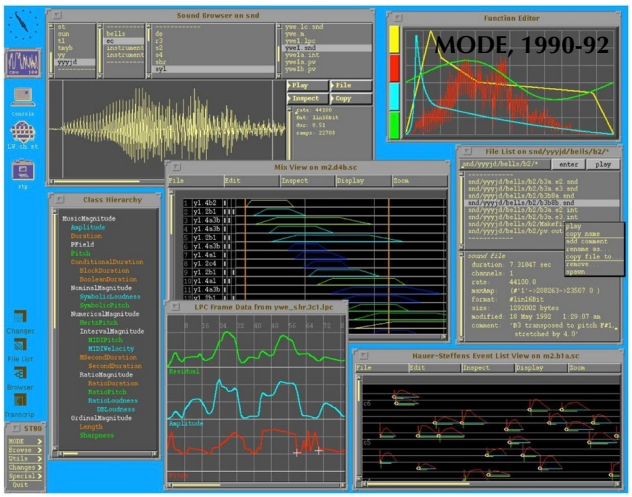
My Background (& biases)



- Training: EE, CS, Music
- HW Development Projects
  - Analog Synths, Eventide
  - Vienna, Salzburg jobs
- Programming
  - Low-level, lots of tool-building, then LISP
  - Visits to MIT and SAIL in 1976/77
  - UNIX & C starting in 1976
  - Graphics starting in 1979

What I Develop

- Complex projects (music, AI, high-level tools)
- Media-rich (A/V, real-time, feature extraction)
- DB-rich (large volumes, complex queries)
- Wide-area distributed
- Cross-platform
- Performance-critical (compute-bound)
- Back-end servers and front-end GUIs
- C/C++, Smalltalk, Python, MATLAB, SuperCollider, SQL



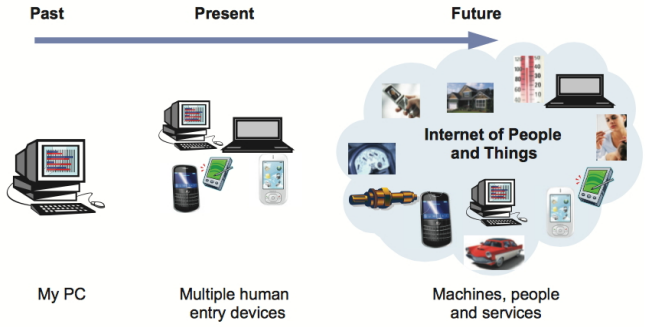
Historical Perspective

Set the stage

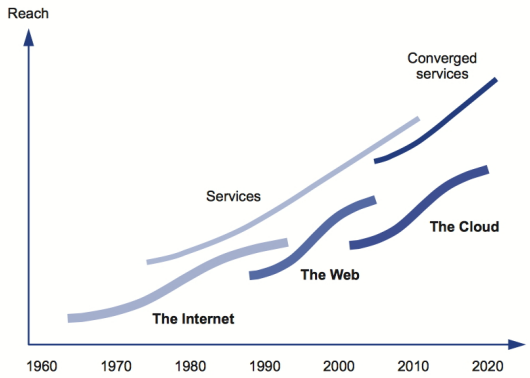
Long-term trends

The Big Picture

UK Future Internet Strategy Group



The Big Picture



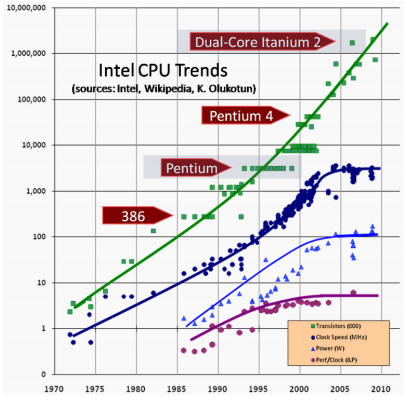
The Big Picture

	Hardware	Language Model	Data Model	Languages	SW Tools	Network	App Model	I/O
Pre-History	Custom	RTL	Core Memory	ASM	None	None	Dedicated	Card Lunch, Line Printer
1960s	Mainframe	Imperative, Algebraic	Pointers, Vectors, Lists	FORTRAN, LISP	Flow charts, Punched cards, print-outs	None	Batch	Control ADCs/ DACs
1970s	Minicomputer, PDP-11	Structured, OO, AI	Data Structures, SQL	Pascal, BASIC, Smalltalk	Shell, compilers, line editors, make	Ethernet LAN, ARPANet, Dial-up uucp	Interactive, R-E-P-Loop	Interactive terminal
1980s	Midi/Micro Computers, VAX, PC, Mac	Structured, Visual	OO Design Patterns	C, VB, ES Shells	TurboPascal, Smalltalk, emacs	ATM WAN, Sockets, CORBA	WIMP EvtLoop	Mouse, MIDI
1990s	Next/SGI/SUN UNIX, PCs	OO	UML, Go4	C++, Java, ObjC	Turbo-family, RatRose UML	WAN, WWW	Client/Server	USB, OSC, Video
2000s	Notebook, Browser, µController	Scripting, Patching, ML	XML, REST	JavaScript, Python, Processing	Turbo-family, Spyder	P2P, AdHoc	Web service, Servlet	FireWire, WiFi, Sensors
2010s	Mobile device, Netbook, Streaming Srvr	SOA	JSON	Dart, Julia, Swift	Turbo-family, Browser	PAN, MAN	App, social network	T-Bolt, 3D Cameras & Printers
Next Big Thing	2nd Digital Hub, *AN Services, Sensors	SOA-like, Brokers	CORBA-like ADT, Service	Decl scripting & service config lang	New Tools, Scripting, Data Shape-change, Service mgmt	BAN, VAN, Streaming LANs, Cloud, Auto-discovery	Service Brokers, A-Discovery, Telepresence, Logging	Context aware Devices/Apps & Services

The Formula

$$\frac{N\,B\,T}{2010s} >> \frac{1980s}{1960s}$$

Moore's Law (again)



#-Xistors

CPU perf

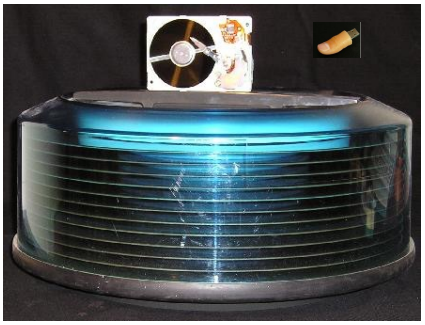
Screen size

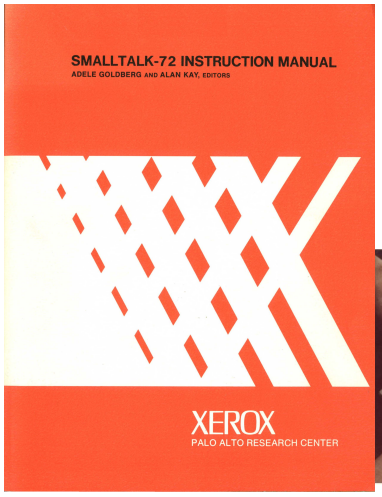
I/O devices

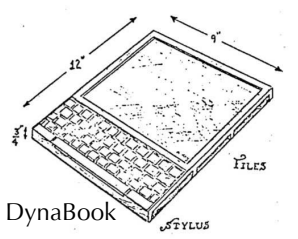
LAN/WAN speeds

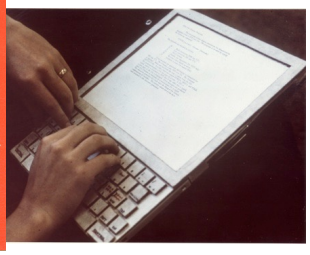
Connectivity

Really?









Some History (1970s)

Xerox Alto (1973)

MIT LM-2 (1976)

Interim Dynabook

Projections

"WAN speed will eventually obviate the need for local storage or computation."

"Storage and computation will be so cheap that you can store/recompute everything of interest locally."

Services

Networks

Tools

Services

What's a computer/platform?

What's an app/service?

What's data?

What's a computer?

- Hardware Platforms: 3 Generations
  - Batch, WIMP, Mobile
- Past Projections
  - Set-top box (media server & WiFi)
  - Thin client (Netbook & cloud apps)
  - Information appliance
  - "Digital Hub"

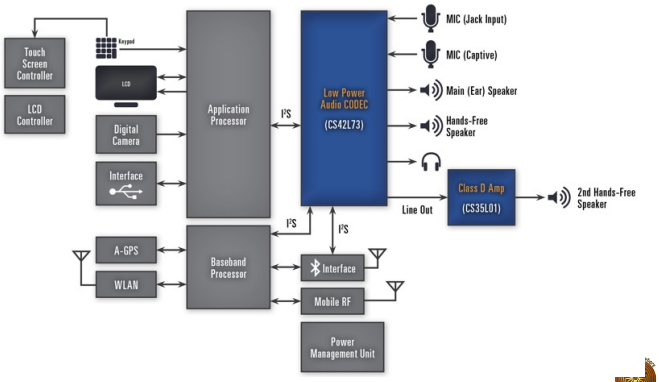
What's the Next-Gen?

- Lap-top, phone, watch, microcontroller, remote control and 1-piece CPUs
- I/O devices determine what you call the appliance (smart home, wearable)
- Delivery/aggregation Platform
  - Netbook, digital hub
- Storage/computation are commodities

Modern Smartphone

- Multi-Core RISC CPU
- Plentiful RAM & Flash
- Graphics & Sensor Co-processors
- 2-4 Radios (cell, WiFi, GPS, BlueTooth)
- Always on, always networked (optional)
- Multi-touch GUI, A/V I/O
- Form factor determined by UI (currently getting larger rather than smaller)

Host du Jour



2nd Digital Hub

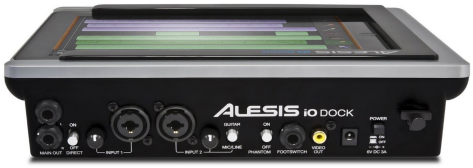
- 2000: Desktop as hub
- Has both computation and storage
- 2015
  - I/O boxes define device
  - Services may run anywhere
  - Return of the co-processor

A/V I/O Examples

For your inner DJ

25

The Better to mix you with…



26

The return of the co-processor!

Out-board device may be larger and costlier than host

27

UAD Satellite: Up to 8 Cores

28

The API

29

Perf -scaling

UAD-2 Instance Count Chart

About This Chart

The following table indicates DSP usage and instance counts for UAD Powered Plug-Ins.
See bottom of page for more details about the chart.

UAD Powered Plug-In	DSP %		SOLO		DUO		QUAD		OCTO	
	Mono	Stereo	Mono	Stereo	Mono	Stereo	Mono	Stereo	Mono	Stereo
4K Buss Compressor	2.8%	3.4%	35	29	70	58	140	116	280	232
4K Channel Strip *	7.4%	11.4%	17	11	34	22	68	44	136	88
Ampex ATR-102 Mastering Tape Recorder	17.6%	29.0%	5	3	10	6	20	12	40	24
AMS RMX16 Digital Reverb	40.6%	41.1%	2	2	4	4	8	8	16	16
API 550A EQ	7.2%	11.7%	13	8	26	16	52	32	104	64
API 560 EQ	9.2%	15.5%	10	6	20	12	40	24	80	48
API Vision Channel Strip *	22.4%	29.7%	4	3	8	6	16	12	32	24
bx_digital V2 EQ & De-Esser	3.4%	4.9%	N/A	20	N/A	40	N/A	80	N/A	160
bx_digital V2 Mono EQ & De-Esser	3.4%	3.8%	29	20	58	40	116	80	232	160
bx_refinement	12.3%	11.9%	7	7	14	14	28	28	56	56
bx_saturator V2	13.2%	12.8%	7	7	14	14	28	28	56	56
bx_tuner	14.7%	15.1%	3	3	6	6	12	12	24	24
Cambridge EQ *	2.2%	3.3%	46	30	92	60	184	120	386	240
Chandler Limited GAV19T Amplifier	40.6%	41%	2	2	4	4	8	8	16	16
Cooper Time Cube	4.5%	4.9%	18	18	36	36	72	72	144	144
CS-1 Channel Strip *	6.0%	6.4%	16	15	32	30	64	60	128	120
Dangerous BAX EQ Master	N/A	3.2%	N/A	31	N/A	62	N/A	124	N/A	248
Dangerous BAX EQ Mix	2.7%	3.1%	37	32	74	64	148	128	296	256
dbx 160 Compressor	2.0%	2.7%	49	36	98	72	196	144	384	288
DM-1 & DM-1L Delay Modulator	1.4%	1.8%	71	54	142	108	284	216	512	432
DreamVerb	11.8%	12.2%	8	8	16	16	32	32	64	64
EL7 FATSO Jr/Sr *	25.6%	40.5%	4	3	8	6	16	12	32	24
elysia• alpha compressor master	35.0%	35.4%	2	2	4	4	8	8	16	16
elysia• alpha compressor mix	31.5%	31.9%	3	3	6	6	12	12	24	24
elysia• mpressor	20.2%	20.6%	4	4	8	8	16	16	32	32
EMT 140 Plate Reverb	14.6%	15.0%	6	6	12	12	24	24	48	48
EMT 250 Electronic Reverberator	6.2%	7.8%	12	11	24	22	48	44	96	88
ENGL E646 VS	43.3%	43.7%	1	1	2	2	4/8**	4/8**	16	16
ENGL E765 RT	42.2%	42.6%	1	1	2	2	4/8**	4/8**	16	16
EP-34 Tape Echo	26.7%	28.1%	3	3	6	6	12	12	24	24
EX-1 EQ/Compressor *	1.9%	2.3%	53	43	106	86	212	172	424	344
Fairchild 660 (Tube Limiter Collection)	13.1%	18.8%	7	5	14	10	28	20	56	40

30

Future IO?

- Multimedia used to mean A/V
  - Camera-/microphone-based input
- Next
  - Multi-touch input
  - 3D scanners/printers (projectors?)
  - Bio/Environ sensors/actuators
  - Ubiquitous computing
    - Sensing rooms, vehicles, etc.

Progress

1990s: "Plug'n'Pray"

2010s: "Shake it again"

Example

2010 UCSB MAT Student Project: Multi-touch "Reactable" (AKA "Surface")

End Product: Slate Raven

Example: My Studio

Example: My Studio

Example: My Studio

What's an app?

Applications vs Apps vs Services

The Promise of WIMP

The Promise of Mobile, Ubiquity, IoThings, Sensors, …

What Hasn't Happened (yet)?

App Categories

Data processing

Report generation

Document (text, spreadsheet) processing

Web browser/content publishing/sites

Social networking site/app

Telepresence-as-app

Life-blogging/monitoring

New App Categories

Data/Content-centric

- Data power tools
- Record, acquire, publish, manage

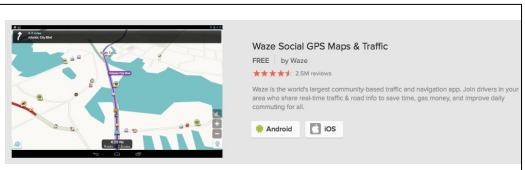
Network-centric

- Telepresence

Community-centric

- Social networks

New Genres



Dynamic Social Networks

Social Network Data Mining (WAZE)

Life (b)logging, Monitoring

Body/environment monitor

Converged services and Brokers

Telepresence-as-app, in-app

TAA

- Skype (why no video walls?)

Social networks are telepresence

Teleoperation (coming soon)

CSCW as/in apps

- Share data, find other users

Multimedia as a Technology Driver

Past
- Storage formats (CD, DVD, DAT)
- Streaming protocols (RTP, CORBA A/V)

Present
- Content-related metadata
- Co-processor API model (GPU, PS3)

Future
- MM data
- Shape change operators & feat. extract.
- Labels vs content

What's your Data?

Then
- Scalars, vector, tables, lists, …
- Tabular databases described in SQL

Now
- Complex dynamic data (in some domains)
- NoSQL
- BigData, Metadata

BigData, Metadata

Access to BigData (local, net)

Metadata in general
- Data about data
- Simple: ID3 tags
- Complex: derived finger-print, bird species ID, annotation of audio life-blog

Data mining: may or may not involve analysis, new metadata generation

Plastics?

No!

Content-derived metadata!

Whither CSCW?

…as in

Computer Support for Collaborative Work

Skype, and email? Really?

FIGURE 2. Meetings were often conducted cross-site.

Introduction: PARC Media Space

- Shared space, not an activity, always "on"
- Allowed
  - planned interactions,
  - chance encounters, and
  - awareness
- Supported cross-site project teams and reporting relationships in existing practices

Offices

Conference rooms

Common areas

Labs

Outside views

CSCW 2006 / Sara Bly

What is was supposed to be…

Figure 4.10 ClearBoard-1 in use

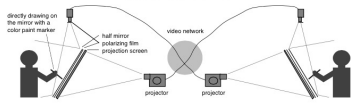

directly drawing on the mirror with a color paint marker
half mirror polarizing film projection screen
video network
projector
projector

Figure 4.11 System architecture of the ClearBoard-1 prototype

Task/Media Analyses

Place

Time	Same	Different
Same	Electronic meeting room	Application sharing
Different	Shift work, team rooms	Email, newsgroups

Figure 5a. Four-Square Map of Groupware Options from Johansen et al (1991).

Place

Time	Same	Different but predictable	Different and unpredictable
Same	Electronic meeting room	Desktop video-conferencing	Multicast events
Different but predictable	Work shifts	Electronic mail	Newsgroups
Different and unpredictable	Team rooms	Collaborative writing	Workflow

Figure 5b. Distinguishing predictable and unpredictable differences in time and place from Grudin and Poltrock (1991).

Detailed Task/Process Taxonomy

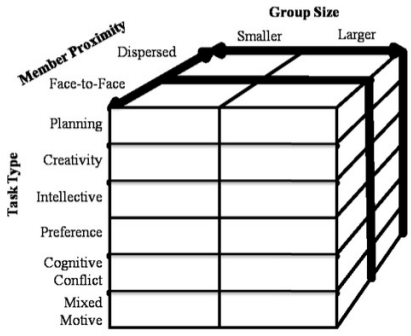


Figure 8. A taxonomy representing group size, member proximity, and task types from DeSanctis & Gallupe (1987).

Social aspects in computing

- Computer-mediated relationships
  - Differences between direct and mediated interaction
  - Social vs real personal networks
- Relationships with devices and services
  - Differences between a programmed device and an individual

First International Congress on

Love and Sex with Robots

November 11th, 2014 | Advances in Computer Entertainment (ACE) 2014, Funchal, Madeira

- Robot Emotions
- Humanoid Robots
- Clone Robots
- Entertainment Robots
- Robot Personalities
- Teledildonics
- Intelligent electronic sex hardware
- Gender Approaches
- Affective Approaches
- Psychological Approaches
- Sociological Approaches
- Roboethics
- Philosophical Approaches

Special Issue on "Ubiquitous Multimedia Systems in Smart Environments"

Interlude I

https://www.youtube.com/watch?v=AkwX4GToDXY

Networks

How do we connect to remote users, services & data stores?

Generations of Technology

Network

- Ethernet LAN, ARPANet, Dial-up uucp
- ATM WAN, Sockets, CORBA
- WAN, WWW
- P2P, AdHoc
- PAN, MAN
- BAN, VAN, Streaming LANs, Cloud, Auto-discovery

Networking

Internet: protocols rather than devices

- Blessing & curse

New Issues

- Sensing servers and streaming devices
- Ubiquity
- Smart appliances

Audio Networking Standards

Audio network technology matrix[1]

Technology	Transport	Transmission scheme	Mixed use networking	Control communications	Topology	Fault tolerance	Distance	Diameter	Network capacity	Latency	Maximum available sampling rate
AES47[1][2]	ATM	Isochronous	Coexists with ATM	Any IP or ATM protocol	Mesh	Provided by ATM	Cat5=100m, MM=2km, SM=70km	Unlimited	Unlimited	125 µs per hop	192 kHz
AES50[3]	Ethernet physical layer[4]	Isochronous or synchronous	dedicated Cat5	5 Mbps Ethernet	Point-to-point	FEC, redundant link	Cat5=100m	Unlimited	48 channels	63 µs	384 kHz and DSD
AES67	Any IP medium	Isochronous	Coexists with other traffic using DiffServ QoS	IP, SIP	Any L2 or IP	Provided by IP	Medium dependent	Unlimited	Unlimited	Minimum possible latency	96 kHz
AudioRail[4]	Ethernet physical layer	Synchronous	Cat5 or fiber	Proprietary	Daisy chain	None	Cat5=100m, MM=2km, SM=70km	Unlimited	32 channels	4.5 µs + 0.25µs per hop	48 kHz (32 ch), 96 kHz (16 ch)
AVB (with IEEE 1722 transport)[5]	Enhanced Ethernet	Isochronous	Coexists with other traffic using AVB QoS and admissions control	Not included	Spanning tree	Provided by 802.1	Cat5=100m, MM=2km, SM=70km	7 hops	Unlimited	2 ms	192 kHz
Avlom Pro64[6]	Ethernet physical layer	synchronous	Dedicated Cat5 and fiber	Proprietary	Daisychain (bidirectional)	Redundant links	Cat5e=120m, MM=2km, SM=70km	9520 km[24]	64 channels	322 µs + 1.34 µs per hop	208 kHz[25]
CobraNet[7]	Ethernet data-link layer	Isochronous	coexists with Ethernet	Ethernet, SNMP, MIDI	Spanning tree	provided by 802.1[26]	Cat5=100m, MM=2km, SM=70km	7 hops, 10 km[24]	Unlimited	1.33 and 5.33 ms	96 kHz
Dante[8]	Any IP medium	Isochronous	Coexists with other traffic using DiffServ QoS	IP, Bonjour	Any L2 or single IP subnet	Provided by 802.1 + redundant link	Cat5=100m, MM=2km, SM=70km	Dependent on latency	700 channels[27]	84 µs or greater[28]	192 kHz
EtherSound ES-100[9]	Ethernet data-link layer	Isochronous	Dedicated Ethernet	Proprietary	star, daisy chain, ring	fault tolerant ring	Cat5=140m, MM=2km, SM=70km	Unlimited	64[29]	84–125 µs + 1.4 µs/node	96 kHz
EtherSound ES-Giga [10]	Ethernet data-link layer	Isochronous	Coexists with Ethernet	Proprietary	Star, daisy chain, ring	Fault tolerant ring	Cat5=140m, MM=600m, SM=70km	Unlimited	512[30]	84–125 µs + 0.5 µs/node	96 kHz
HyperMAC	Gigabit Ethernet	Isochronous	Dedicated Cat5, Cat6, or fiber	100 Mbps+ Ethernet	Point-to-point	Redundant link	Cat6=100m, MM=500m, SM=10km	Unlimited	384+ channels	63 µs	384 kHz and DSD
Livewire[11]	Any IP medium	Isochronous	Coexists with Ethernet	Ethernet, HTTP, XML	Any L2 or IP	Provided by 802.1[31]	Cat5=100m, MM=2km, SM=70km	Unlimited	32760 channels	0.75 ms	48 kHz
mLAN[12]	IEEE-1394	Isochronous	Coexists with IEEE-1394	IEEE-1394, MIDI	Tree	Provided by 1394b	1394 cable (2 power, 4 signal): 4.5m	100 m	63 devices (800 Mbps)	354.17 µs	192 kHz[32]
Optocore[13]	Dedicated fiber	Synchronous	Dedicated Cat5/fiber	Proprietary	Ring	Redundant ring	MM=700m, SM=110 km	Unlimited	512 channels at 48 kHz	41.6 µs	96 kHz
Q-LAN[14]	IP over Gigabit Ethernet	Isochronous	Coexists with other traffic using DiffServ QoS	IP, HTTP, XML	Any L2 or IP	802.1, redundant link, IP routing	Cat5=100m, MM=550m, SM=10km	7 hops or 35 km	Unlimited	1 ms	48 kHz
Rocknet[15]	Ethernet physical Layer	Isochronous	Dedicated Cat5/fiber	Proprietary	Ring	Redundant ring	Cat5e=150 m, MM=2 km, SM=20 km	10 km max, 99 devices	160 channels (48 kHz/24-bit)	400 µs @ 48 kHz	96 kHz
UMAN	IEEE 1394 and Ethernet AVB[33]	Isochronous and asynchronous	coexists with Ethernet	IP-based XFN	daisy chain in ring, tree, or star (with hubs)	fault tolerant ring, device redundancy	Cat5e=50m, Cat6=75 m, MM=1 km, SM=>2 km	Unlimited	400 channels (48 kHz/24bit)[34]	354 µs + 125 µs per hop[35]	192 kHz

Networking Issues

Sensor, mesh and [vehicular, personal, body, room] wireless networks

Middleware services for wireless, mobile and multimedia networks

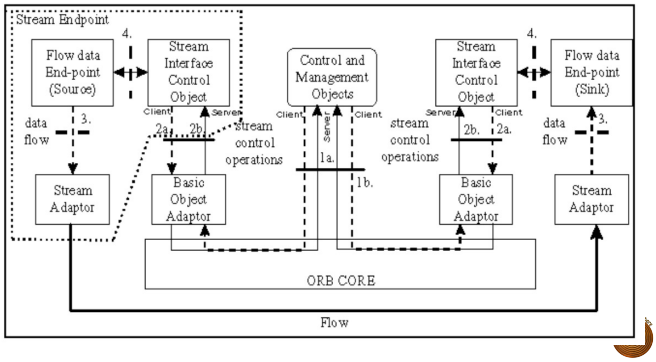
Handoff and mobility management

Software defined wireless networks

All the pretty protocols…

Networking Software Support

CORBA A/V Stream Architecture



How to fix it

Auto-discovery

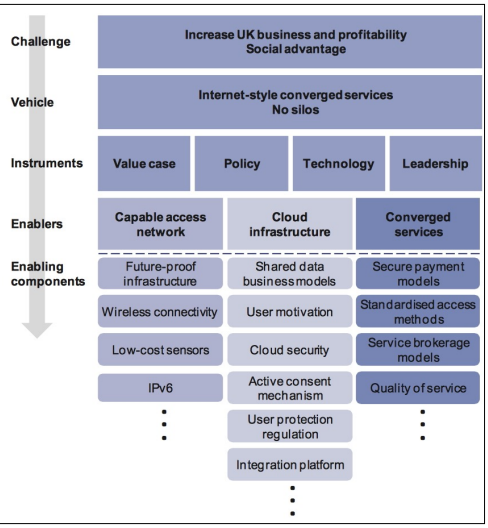
- CORBA
- OSC

Service description (reflection, API delivery)

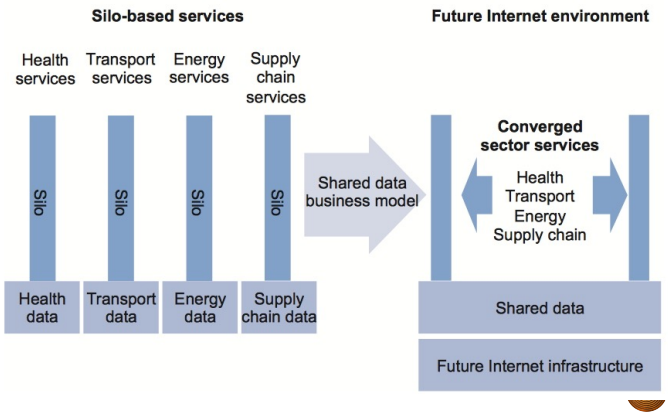
"Fail ever better"

See Apple's Hand-off

Next-Gen Networks



Silos Disappear



Status of Enabling Technologies

Table 1 Status of enabling components for a capable access mechanism

Enabling factor	Level of implementation	State of technology	More R&D required?	Issues
Future-proof Infrastructure	🔴	🟡	✓	• Investment required to deliver an advanced capability
Wireless connectivity	🟡	🔴	✓	• Resolution of spectrum issues for LTE[4] and improving multiband RF front-end components to utilise spectrum more efficiently
Low-cost sensors	🟡	🟡	✓	• Development of multipurpose low-cost sensor arrays • Sensor power • Standards for wireless communications
IPv6	🟡	🟢		• Identify potential opportunities

Key: 🔴 = extensive action required, but issues not yet understood
🟡 = some action required, but issues are known
🟢 = no action required

Status of Cloud

Table 3 Status of enabling components for cloud infrastructure

Enabling factor	Level of implementation	State of technology	More R&D required?	Issues
Shared data business models	🟡	🟢		• Now a question of application design
User motivation	🔴	🔴	✓	• Confidence • Psychological and value case
Cloud security	🟡	🟡	✓	• Security across organisational and national boundaries
Active consent mechanism	🔴	🟡	✓	• Legal and regulatory issues coupled with R&D into more robust and transparent methodologies
User protection regulation	🔴	🟡	✓	• Regulation at UK, EU and global level
Integration platform	🔴	🟡	✓	• Build platforms that will allow applications to be created and marketed (e.g. Apple)

Key: 🔴 = extensive action required, but issues not yet understood
🟡 = some action required, but issues are known
🟢 = no action required

Tools

How is software described and delivered?

Software Development

Language History from the Big Picture

On-going debates about languages and tools

What impacts developer productivity?

Programming Languages

- Maximal (complex, multi-paradigm, kitchen sink)
  - ALGOL
  - PL/I
  - CommonLISP
  - ADA
  - C++
  - Scripting (?)
- Minimal (simple, single paradigm)
  - LISP
  - PASCAL
  - Smalltalk
  - Lua
  - MathLangs
  - APL, FP, Prolog, ...

On-going Debates

- Language Issues
  - Uniform vs hybrid OO (is everything an object?)
  - Kinds of sharing of state and behavior
    - Classes vs prototypes
    - Single/multiple inheritance (composition, traits)
    - Interfaces, protocols, abstract classes
  - Type declarations (vs type inferencing)
    - Performance/reusability/LoE trade-offs
  - Preferred syntax (many variants)
  - Coding style and enforcement
- Library/Tool/Environment issues?

The Perfect Language (IMHO)

- Truly minimal syntax
  - One best way to write any given expression
  - Keyword syntax (rather than algebraic syntax)
  - Very few reserved words (known to compiler)
  - Nouns and verbs, no adjectives
- Everything is an object
- Objects only interact by message-passing
- Constructors are class messages
- Control structures are messages (extensible)
- Objects and classes are reflective
  - isKindOf:, understands:, allInstances, someInstance

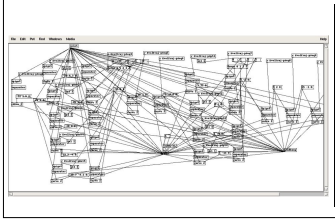
Visual Languages

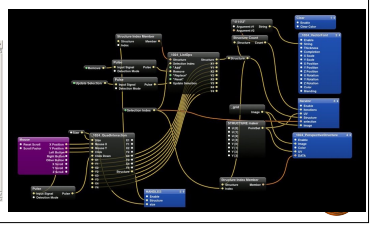
"Block diagram compilers"

PD, QuartzComposer

Potential and Challenges

Use as "scripting" languages





What's the Problem?

Language Complexity

Learning curve issues

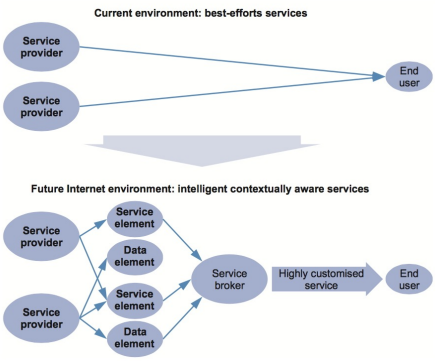
Compiler/optimizer issues

Low reusability

Libraries not standardized

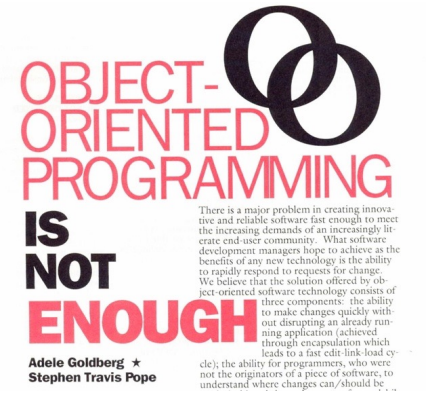
Poor tools for finding things

Service Combination



How to facilitate this?

Programming Today


OBJECT-ORIENTED PROGRAMMING IS NOT ENOUGH

Adele Goldberg ★ Stephen Travis Pope

There is a major problem in creating innovative and reliable software fast enough to meet the increasing demands of an increasingly literate end-user community. What software development managers hope to achieve as the benefits of any new technology is the ability to rapidly respond to requests for change. We believe that the solution offered by object-oriented software technology consists of three components: the ability to make changes quickly without disrupting an already running application (achieved through encapsulation which leads to a fast edit-link-load cycle); the ability for programmers, who were not the originators of a piece of software, to understand where changes can/should be

I Deserve Better than a C+ !

OO Is Not Enough!

Programmer Productivity impacted by:

- Language (simple, expressive)
- Libraries (browsing, reuse)
- Tools (creation, debugging, refactoring)
- Methods (A/D, Project Mgmnt)

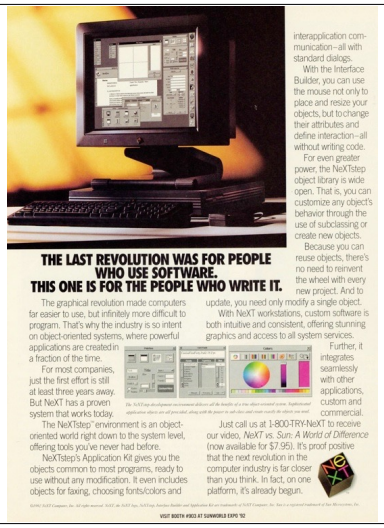
The State of SW Tools

- Flow charts, Punched cards, print-outs
- Shell, compilers, line editors, make
- TurboPascal, Smalltalk, emacs
- RatRose UML
- Spyder
- Smalltalk Browser
- New Tools
  - Scripting, Data Shape-change, Machine learning, Service mgmnt

Summary (spoiler alert)

THE LAST REVOLUTION WAS FOR PEOPLE WHO USE SOFTWARE.
THIS ONE IS FOR THE PEOPLE WHO WRITE IT.

NeXT ad from 1992


THE LAST REVOLUTION WAS FOR PEOPLE WHO USE SOFTWARE.
THIS ONE IS FOR THE PEOPLE WHO WRITE IT.

The graphical revolution made computers far easier to use, but infinitely more difficult to program. That's why the industry is so intent on object-oriented systems, where powerful applications are created in a fraction of the time.

For most companies, just the first effort is still at least three years away. But NeXT has a proven system that works today.

The NeXTstep™ environment is an object-oriented world right down to the system level, offering tools you've never had before.

NeXTstep's Application Kit gives you the objects common to most programs, ready to use without any modification. It even includes objects for faxing, choosing fonts/colors and interapplication communication—all with standard dialogs.

With the Interface Builder, you can use the mouse not only to place and resize your objects, but to change their attributes and define interaction—all without writing code.

For even greater power, the NeXTstep object library is wide open. That is, you can customize any object's behavior through the use of subclassing or create new objects.

Because you can reuse objects, there's no need to reinvent the wheel with every new project. And to update, you need only modify a single object.

With NeXT workstations, custom software is both intuitive and consistent, offering stunning graphics and access to all system services.

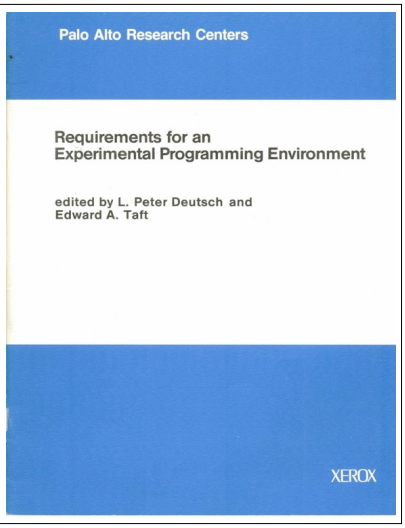
Further, it integrates seamlessly with other applications, custom and commercial.

Just call us at 1-800-TRY-NeXT to receive our video, *NeXT vs. Sun: A World of Difference* (now available for $7.95). It's proof positive that the next revolution in the computer industry is far closer than you think. In fact, on one platform, it's already begun.

Requirements Analysis

- Great Overviews of the 1980s
  - Why Pascal is Not My Favorite Programming Language (BWK, 1981)
  - Requirements for an Experimental Programming Environment (LPD, 1982)
  - The Organization of Expert Systems (1982)
  - An OO Approach to Multimedia Databases (1986)
  - Programmer as Reader (AG, 1986)
  - Requirements for an Experimentation Environment for Composers (STP, 1988)

Req_Exp_Prog_Env



Req_Exp_Prog_Env (lang)

Virtual machine / programming language

(L1) Large virtual address space (≥24 bits)
(L2) Direct addressing for files
(L2a) Segmenting
(L2b) An enormous virtual address space (≥48 bits)
(L3) * Well-integrated access to large, robust data bases
(L4) Memory management—object/page swapping
(L5) * Object management—garbage collection, reference counting
(L6) Some support for interrupts
(L7) * Adequate exceptional condition handling
(L8) User access to the machine's capability for packed data
(L9) * Program-manipulable representation of programs
(L10) * Run-time availability of all information derivable from source program (e.g., names, types, scopes)
(L11) Statically checked type system
(L12) * Self-typing data (*a la* Lisp and Smalltalk), run-time type system
(L13) Encapsulation/protection mechanisms (scopes, classes, import/export rules)
(L14) Abstraction mechanisms; explicit notion of "interface"
(L15) * Non-hierarchical control (coroutines, backtracking)
(L16) Adequate run-time efficiency
(L17) * Inter-language communication
(L18) Uniform screen management
(L19) * Inheritance/defaulting (e.g., Smalltalk subclassing; difficulty depends a lot on how much it has to do)
(L20) * Ability to extend language (e.g., operator overloading)
(L21) * Ability to create fully integrated local sublanguages
(L22) User access to the machine's capability for multi-precision arithmetic

Req_Exp_Prog_Env (L2)

(L23) Good facilities for processes, monitors, interrupts
(L24) * Simple, unambiguous syntax (including infix notation)
(L25) Control over importation of names
(L26) User packages as "first-class citizens"
(L27) * Closures
(L28) Full-scale inter-language communication
(L29) User microprogramming
(L30) Clean data and control trapping mechanisms
(L31) * "Good" exceptional condition handling

Req_Exp_Prog_Env (Tools)

Tools

(T1) * Fast turnaround for minor program changes (less than 5 seconds)
(T2) * Compiler/interpreter available with low overhead at run time
(T3) * Cross-reference/annotation capability
(T4) Prettyprinter
(T5) Consistent compilation
(T6) * Version control
(T7) * Librarian, program-oriented filing system (including Browser)
(T8) Source-language debugger
(T9) Dynamic measurement facilities
(T10) Checkpoint, establishing a protected environment
(T11) History and undoing
(T12) * Editor integrated with language system
(T13) More optimizing compiler if user willing to bind more tightly—with full compatibility
(T14) Aids for incremental development (stubs, outstanding task list)
(T15) Regression testing system
(T16) Random testing aids
(T17) (high capability) Masterscope
(T18) Access to on-line documentation (Helpsys)
(T19) Static analyzers: verifier, performance predictor

Req_Exp_Prog_Env (Libs)

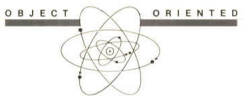
Packages

(P1) Text objects and images
(P2) Line objects and images
(P3) Scanned (bitmap) objects and images
(P4) * Formatted document files
(P5) * More elaborate screen management
(P6) Remote file storage
(P7) Small data base manager
(P8) Message transmission system
(P9) * Remote procedure call
(P10) Event logging
(P11) Background processing
(P12) Generalized cache
(P13) Document editing
(P14) Forms
(P15) Menus and other standard user interfaces
(P16) History lists
(P17) User access to full bandwidth of disk
(P18) (English) dictionary service
(P19) Teleconferencing
(P20) * Audio
(P21) User access to full bandwidth of networks

Req_Exp_Prog_Env

Other

(X1) Adequate reference documentation
(X2) * "Efficient" interface for experts
(X3) Uniformity in command interface
(X4) * "Self-teaching" interface for beginners
(X5) Good introductory documentation



Programmer as Reader

- Insight: Software engineers spend more time reading than writing; therefore code reading tools are really important.
- Later insight: Code *rewriting, refactoring, …*
- "Reading and writing are partners"
  - Should use same/similar tools
  - E.g., not emacs and google

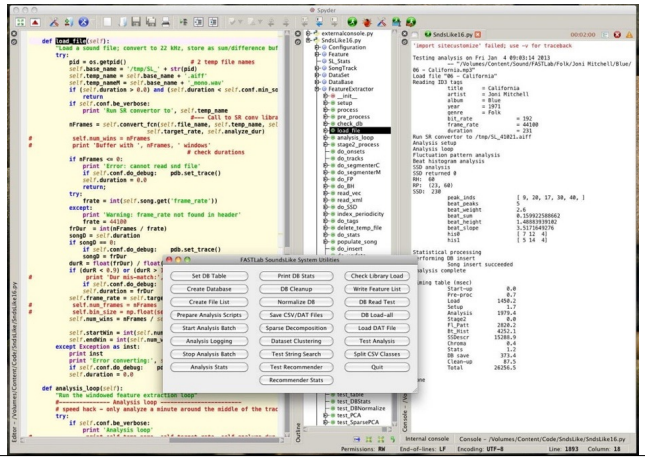
Programmer as Reader

- Rules of Reuse:
  1: If you can't find it, you can't reuse it
  2: If you don't trust it, you won't reuse it
- Ease of learning via high-level concepts
  - As in real life
- Encourage good design & coding style by use of templates
- Design of Smalltalk Browsers
  - Find, test, reuse

Summary

- Smart people have thought long and hard about how to improve programmer productivity and code quality
  - Language issues
  - Library issues
  - Development tool issues
  - Operating system and environment
- Support for OOP & rapid prototyping and incremental refinement (refactoring, agileP)

Example: Spyder



Smalltalk Libraries

- Single root class (Object)
- All classes can be extended (!)
- Heavy use of polymorphism
- Mostly single inheritance
  - Use composition and forwarding by default
  - Other sharing mechanisms (species, MI, generic behaviors, traits, interfaces, passing blocks, etc.)
  - "Core" = ~1000 classes; VW 7.9 = ~3300 classes

Smalltalk Libraries

- Protocol of Object
  - Messages understood by all objects
- Important high-level abstractions
  - Magnitudes (number, point, date, character)
  - Collections (array, string, set, dictionary)
  - Streams (file, socket, producer, server)
  - Model, View, Controller classes
  - Graphics classes
  - Event classes

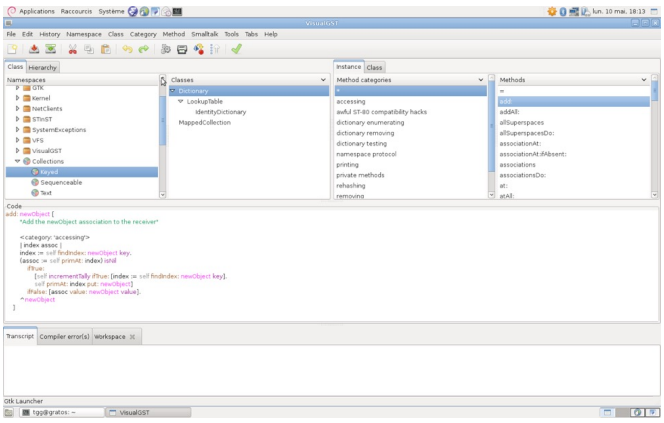
Smalltalk IDE

- Browsers
  - Browse collections of objects (e.g., classes)
  - Organization of set
  - Operations on inner nodes & leaf nodes
- Inspectors
  - Low-level access to an object's properties
  - May violate encapsulation
- Editors
  - Customized inspector
  - May add custom GUI (V/C)

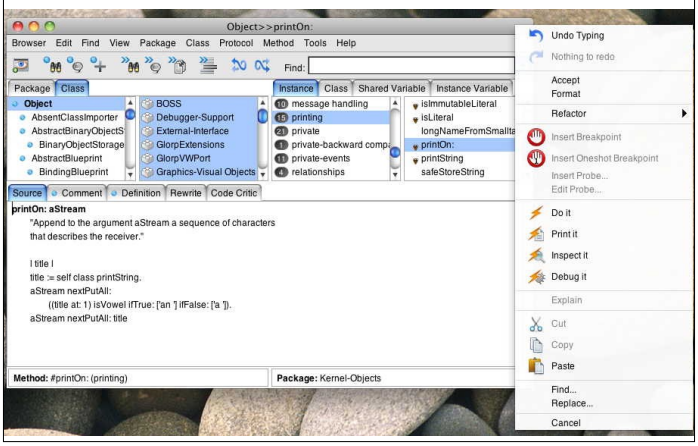
Code Browsers

- Uniform access to <u>all</u> system source code in a hypertext browser with full-text search
- Unit of change (compilation) is a method
- Many kinds
  - Package, Namespace, Hierarchy, Refactoring
  - Cross-referencing (reflection & code DB)
- Code editing aids
  - Formatter, spelling, auto-complete, explainer
  - Templates for classes, methods, documentation
- Standard system browser layout

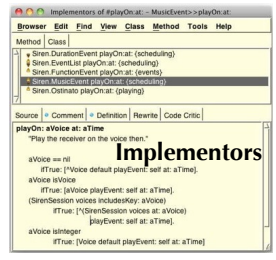
System Browser (GST)

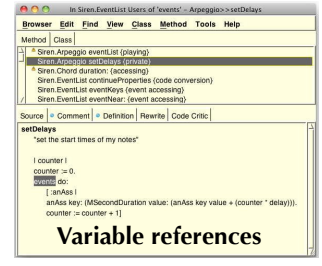


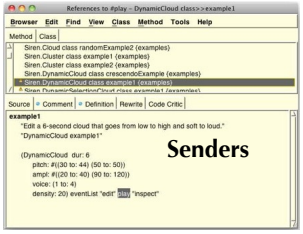
Class Object

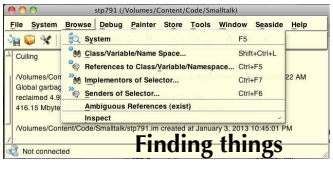


Browsers

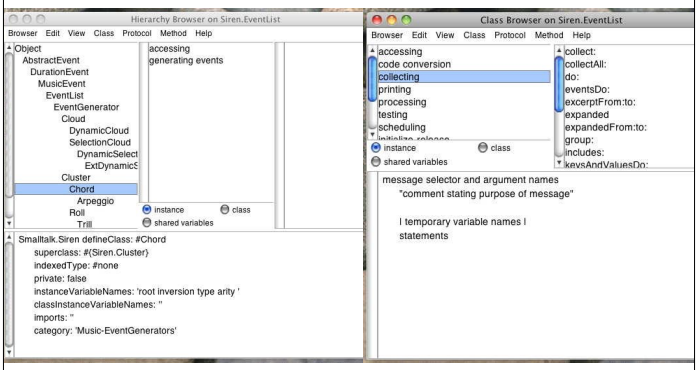




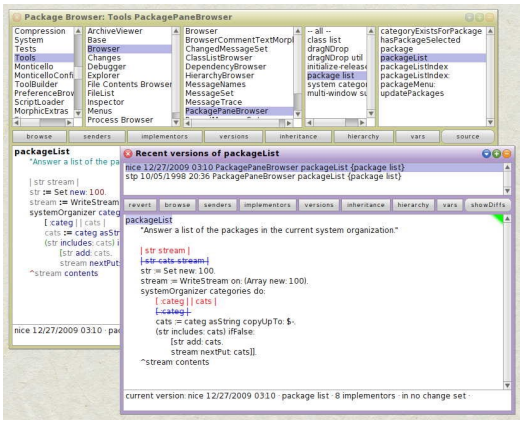




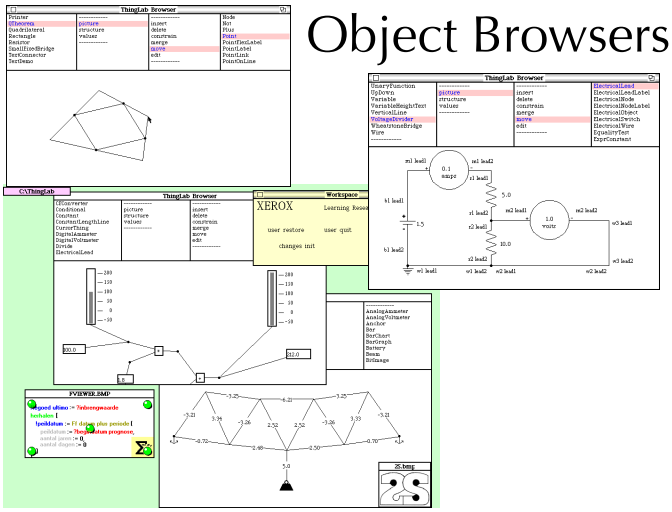
Browser Templates



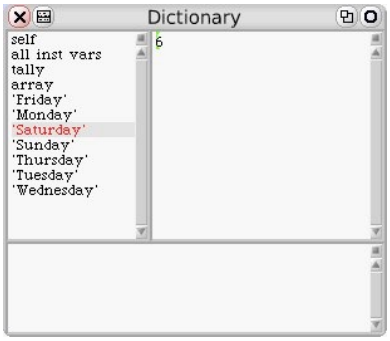
Code History


Package Browser: Tools PackagePaneBrowser
Recent versions of packageList
packageList
"Answer a list of the packages in the current system organization."
current version: nice 12/27/2009 03:10 · package list · 8 implementors · in no change set ·

Object Browsers

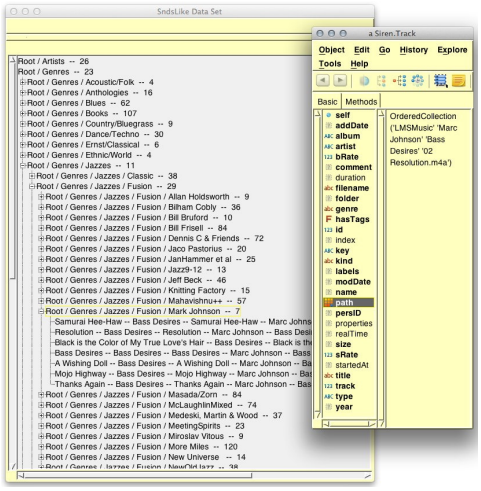


Inspectors

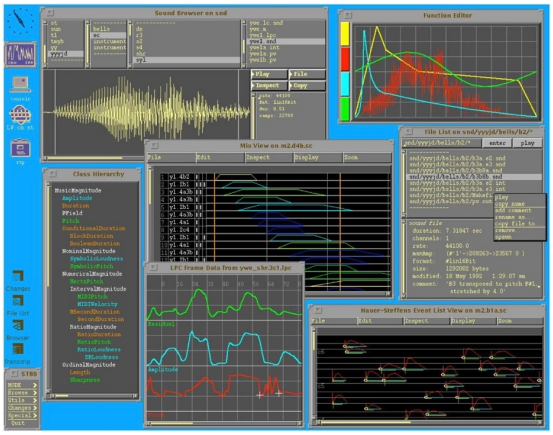


The “soldering iron” - may violate encapsulation

Song DB Browser/ Inspector


SndsLike Data Set
a Siren.Track
Root / Artists -- 26
Root / Genres -- 23
Root / Genres / Acoustic/Folk -- 4
Root / Genres / Anthologies -- 16
Root / Genres / Blues -- 62
Root / Genres / Books -- 107
Root / Genres / Country/Bluegrass -- 9
Root / Genres / Dance/Techno -- 30
Root / Genres / Ernst/Classical -- 6
Root / Genres / Ethnic/World -- 4
Root / Genres / Jazzes -- 11
Root / Genres / Jazzes / Classic -- 38
Root / Genres / Jazzes / Fusion -- 29
Root / Genres / Jazzes / Fusion / Allan Holdsworth -- 9
Root / Genres / Jazzes / Fusion / Bitham Cobly -- 36
Root / Genres / Jazzes / Fusion / Bill Bruford -- 10
Root / Genres / Jazzes / Fusion / Bill Frisell -- 84
Root / Genres / Jazzes / Fusion / Dennis C & Friends -- 72
Root / Genres / Jazzes / Fusion / Jaco Pastorius -- 20
Root / Genres / Jazzes / Fusion / JanHammer et al -- 25
Root / Genres / Jazzes / Fusion / Jazz9-12 -- 13
Root / Genres / Jazzes / Fusion / Jeff Beck -- 46
Root / Genres / Jazzes / Fusion / Knitting Factory -- 15
Root / Genres / Jazzes / Fusion / Mahavishnu++ -- 57
Root / Genres / Jazzes / Fusion / Mark Johnson -- 7
Samurai Hee-Haw -- Bass Desires -- Samurai Hee-Haw -- Marc Johnson
Resolution -- Bass Desires -- Resolution -- Marc Johnson -- Bass Desires
Black is the Color of My True Love's Hair -- Bass Desires -- Black is the
Bass Desires -- Bass Desires -- Bass Desires -- Marc Johnson -- Bass
A Wishing Doll -- Bass Desires -- A Wishing Doll -- Marc Johnson -- Bass
Mojo Highway -- Bass Desires -- Mojo Highway -- Marc Johnson -- Bass
Thanks Again -- Bass Desires -- Thanks Again -- Marc Johnson -- Bass
Root / Genres / Jazzes / Fusion / Masada/Zorn -- 84
Root / Genres / Jazzes / Fusion / McLaughlinMixed -- 74
Root / Genres / Jazzes / Fusion / Medeski, Martin & Wood -- 37
Root / Genres / Jazzes / Fusion / MeetingSpirits -- 23
Root / Genres / Jazzes / Fusion / Miroslav Vitous -- 9
Root / Genres / Jazzes / Fusion / More Miles -- 120
Root / Genres / Jazzes / Fusion / New Universe -- 14
Object Edit Go History Explore
Tools Help
Basic Methods
self
addDate
album
artist
bRate
comment
duration
filename
folder
genre
hasTags
id
index
key
kind
labels
modDate
name
path
persID
properties
realTime
size
sRate
startedAt
title
track
type
year
OrderedCollection ('LMSMusic' 'Marc Johnson' 'Bass Desires' '02 Resolution.m4a')

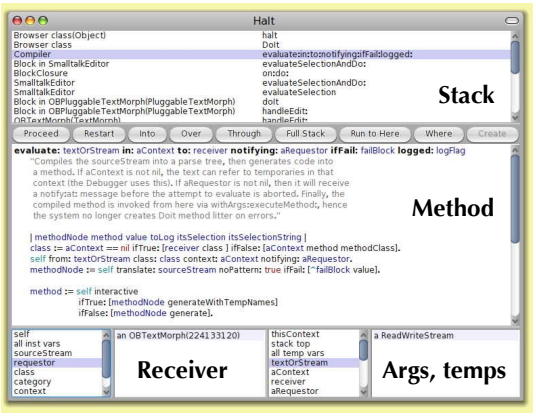
Editors



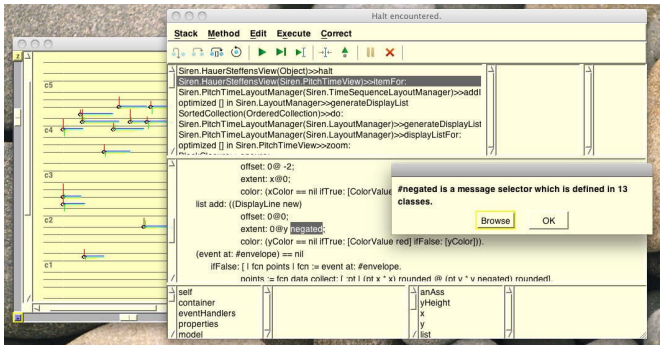
Debuggers

- Basically a method-list browser where the method list is the stack of some thread
- Adds inspectors on the receiver object, method arguments and temporary objects
- Allows full browser-style cross-referencing
- Supports fix&continue - you can recompile a method in the middle of the stack and continue from there

Debugger

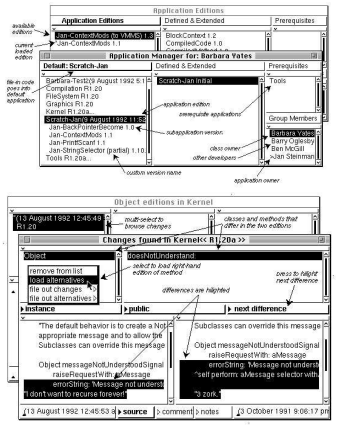


Debugging a View's Display Method


Halt encountered.
Stack Method Edit Execute Correct
#negated is a message selector which is defined in 13 classes.
Browse
OK

Code Management & CASE

- Supporting multi-person teams
- Projects and changes
  - Change list tool
- Namespaces
- Configurations
- Package managers



Smalltalk and OOA/D

- OO Design Patterns
  - Most existed in ST before the Go4 book
- Early OOA/D methods
  - OBA, CRC Cards
- UML and formal OO Methods
  - Notations
  - Processes
- Domain-specific languages
- XP, Agile methods

Smalltalk Virtual Image

Save a file of a running app with all libs, tools, running processes, etc.

Virtual Machine loads obj network at start-up

Like a .jar in Java or MATLAB workspace

Language-independent IDEs

- Examples
  - XCode, VisualStudio, Eclipse, NetBeans
- All are file-based
  - Integrate with SVN, Mercurial, git, etc.
- None have code queries (who sends this?)
- Weak debuggers (fix&continue)
- Programmer as reader
  - Finding, testing, reusing
  - Refining, refactoring
- Smalltalk IDE for julia, python or swift?

So, why do I still program in Smalltalk?

- Realtor's answer:
  - Productivity, productivity, productivity
  - Productivity, reliability, malleability
- CS Answer
  - Development cost
    - Reliability, maintainability
    - Portability (!)
    - Stability (!)
  - MM libraries
  - Powerful, open-source tools

Interlude II

http://vimeo.com/92867679

Dynamic Open-Source Tools

"Any sufficiently complex app becomes a domain-specific language"

"Any sufficiently complex framework becomes a domain-specific tool kit"

Writing NextGen Apps

Embrace the change (post-WIMP)

Social networks in apps

App as "converged service" configuration

Address the platform

Sensors, MM I/O, networking with other users (LAN, MAN, WAN, GPS, …)

GUI -> user experience

A Call to Action

What features should we all demand of our software development tools?

A Call to Action

- Language demands:
  - Uniform OO model
    - Everything is an object, including: #s, classes, threads, closures, Booleans, message-sends, error conditions, ...
    - Class messages (not just for constructors!)
    - Many ways of sharing: SI/MI, traits, interfaces
  - Simple (unambiguous) syntax (1-page BNF)
  - Go easy on the adjectives, reserved words and punctuation
  - Straight-forward to express other paradigms (grammars, rules, logic, ...) in the same syntax

A Call to Action

- Language demands:
  - Keyword messages
    - Avoid f(a, b, c, d, e)
    - e.g., aGraphicsContext copyArea: aShape from: aGraphicsContext sourceOffsetX: srcOffsetX y: srcOffsetY destinationOffsetX: destOffsetX y: destOffsetY
  - Extensible (OO) control structures
  - Flexible exception-handling
  - Support for concurrency (would be nice)
  - Explicit style guidelines (obeyed by libraries)
  - Get it right (V3) before you have an installed base!

A Call to Action

- Library demands:
  - Consistent, comprehensive & small class libraries
    - Built for programmer-as-reader (doc, examples)
    - Same design, naming and formatting style throughout
  - OO design patterns
    - Standard and new
  - Small set of high-level abstractions
    - E.g., Magnitude, Collection, Stream, Event, DisplayList
  - Powerful mid-level abstractions
    - E.g., Number, SequenceableCollection, BufferedStream
  - Minimal concrete classes (opt. constructor-only)
    - E.g., Float, Array, FileStream, Chord

A Call to Action

- Library demands:
  - MVC classes - widgets and app models
    - High-level widgets, document model, browser
  - Multimedia support (incl. scheduler)
    - Graphical models
    - Sound/music models
    - Others (gesture)
  - (R/OO)DB support, O/R mapping, NoSQL/JSON
  - Proxies and distributed systems
  - Parser-generator and compiler framework
  - Compiler available at run-time

A Call to Action

- IDE demands (coding):
  - Code lives in hypertext document DB
  - Programmer-as-reader code browsers with full-text search, cross-ref, explainers, formatting, spell checker, auto-complete, ...
  - Maintain method history & diffs
    - Easy roll-back of changes
  - Unit of change is a single method
  - Support for multi-person team programming

A Call to Action

- IDE demands:
  - Inspector framework for operating on objects
  - Debuggers look like code browsers
  - Fix&continue source code debuggers
  - Workspace (VI) development/delivery
    - Dynamic object network in RAM
    - Object DB queries (allInstances)
  - Tools are open-source and extensible
    - Easy to customize browsers
    - Easy to build new inspectors & editors
  - Tools work; tools scale

If I got my wish...

- Merge Pharo and NewSpeak
  - Classes and prototypes
- Add
  - MI, interfaces, slang
- Add easy DLLCC (a la ctypes in Python)
- Add stand-alone mode (Bootable ST)
- Port to:
  - Desktop OSs (as process or stand-alone)
  - Raspberry Pi, Arduino, mobile platforms

Conclusions

- Object-Oriented is not Enough!
- The computer revolution ...
  - hasn't happened yet.

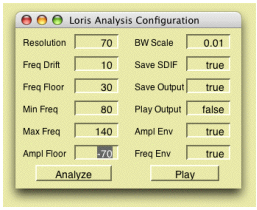
What Comes After That?

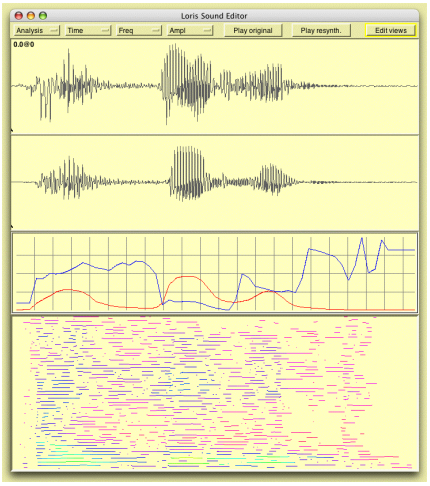
Service Issues
- App combination, reuse
- New app categories

Network Issues
- Reflective, service-oriented networks
- Integrate remote data, big data, metadata
- Connect the user to other users

Tool Issues
- How to develop and reuse high-level components?

Why am I doing this?





Options

Q&A

More Demo Videos (NIME)

Music Music & Videos

IDE Slides

Smalltalk Demo

Thank You !